The Parametric Closure Problem

David Eppstein

Computer Science Department, Univ. of California, Irvine, USA

Abstract. We define the *parametric closure problem*, in which the input is a partially ordered set whose elements have linearly varying weights and the goal is to compute the sequence of minimum-weight lower sets of the partial order as the weights vary. We give polynomial time solutions to many important special cases of this problem including semiorders, reachability orders of bounded-treewidth graphs, partial orders of bounded width, and series-parallel partial orders. Our result for series-parallel orders provides a significant generalization of a previous result of Carlson and Eppstein on bicriterion subtree problems.

1 Introduction

Parametric optimization problems are a variation on classical combinatorial optimization problems such as shortest paths or minimum spanning trees, in which the input weights are not fixed numbers, but vary as functions of a parameter. Different parameter settings will give different weights and different optimal solutions; the goal is to list these solutions and the intervals of parameter values within which they are optimal. As a simple example, consider maintaining the minimum of n input values as a parameter controlling these values changes. This parametric minimum problem asks for the *lower envelope* of a collection of input functions; for linear functions this is equivalent by projective duality to a planar convex hull [25], and can be constructed in time $O(n \log n)$; more general function classes such as piecewise-polynomial functions also have efficient lower-envelope algorithms [16]. The parametric minimum spanning tree problem (with linear edge weights) has polynomially many solutions that can be constructed in polynomial time [1, 12, 14]; the parametric shortest path problem has a number of solutions and running time that are exponential in $\log^2 n$ on n-vertex graphs [7].

As well as the obvious applications of this formulation to real-world problems with time-varying but predictable data (such as rush-hour route planning), parametric optimization problems have another class of applications, to *bicriterion optimization*. In bicriterion problems, each input has two numbers associated with it, that can be summed over the elements of a candidate solution. For instance, these two numbers might be the x and y coordinates of points in the plane, the mean and variance of a normal distribution, an initial investment cost and expected profit of a business opportunity, or the cost and log-likelihood of failure of a communications link. The goal is to find a solution that optimizes a nonlinear combination of these two sums of values, such as the distance from the origin, probability of exceeding a given threshold, percentage return on investment (the

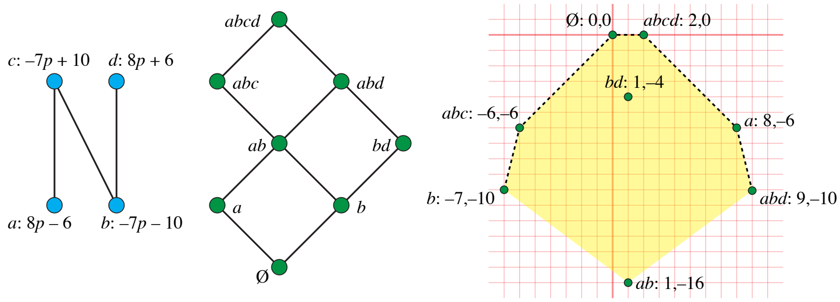

Fig. 1. An instance of the parametric closure problem. Left: The Hasse diagram of a partially ordered set N of four elements, each with a weight that varies linearly with a parameter p. Center: The distributive lattice $\mathsf{lower}(N)$ of lower sets of N. Right: The point set $\mathsf{project}(\mathsf{lower}(N))$ and its convex hull. The upper hull (dashed) gives in left-to-right order the sequence of six distinct maximum-weight closures as the parameter p varies continuously from $-\infty$ to $+\infty$.

ratio of total profit to total investment cost), or cost-reliability ratio. Many natural bicriterion problems can be expressed as finding the maximum of a quasiconvex function of the two sums (a function whose lower level sets are convex sets) or equivalently as finding the minimum of a quasiconcave function of the two sums. When this is the case, the optimal solution can always be obtained as one of the solutions of a parametric problem, defined by re-interpreting the two numbers associated with each input element as the slope and y-intercept of a linear parametric function [6, 21]. In this way, any algorithm for solving a parametric optimization problem can also be used to solve bicriterion versions of the same type of optimization problem.

In this paper we formulate and provide the first study of the *parametric closure problem*, the natural parametric variant of a classical optimization problem, the *maximum closure problem* [18, 28]. A closure in a directed graph is a subset of vertices such that all edges from a vertex in the subset go to another vertex in the subset; the maximum closure problem is the problem of finding the highest-weight closure in a vertex-weighted graph. Equivalently we seek the highest-weight *lower set* of a weighted partial order, where a lower set is a subset of the elements of a partial order such that if $x < y$ in the order, and y belongs to the subset, then x also belongs to the subset. Applications of this problem include open pit mining [23], military attack planning [27], freight depot placement [3, 30], scheduling with precedence constraints [8, 22], image segmentation [2, 15], stable marriage with maximum satisfaction [19], and treemap construction in information visualization [5]. Maximum closures can be found in polynomial time by a reduction to maximum flow [17, 28] or by direct algorithms [13].

In the parametric closure problem, we assign weights to the vertices of a directed graph (or the elements of a partial order) that vary linearly as functions of a parameter, and we seek the closures (or lower sets) that have maximum weight as the parameter varies. As described above, an algorithm for this problem

can also solve bicriterion closure problems of maximizing a quasiconvex function (or minimizing a quasiconcave function) of two sums of values. Although we have not been able to resolve the complexity of this problem in the general case, we prove near-linear or polynomial complexity for several important special cases of the parametric closure problem.

We do not know of previous work on the general parametric closure problem, but two previous papers can be seen in retrospect as solving special cases. Lawler [22] studied scheduling to minimize weighted completion time with precedence constraints. He used the closure that maximizes the ratio x/y of the priority x and processing time y of a job or set of jobs to decompose instances of this problem into smaller subproblems. As Lawler showed, the optimal closure can be found in polynomial time by a binary search where each step involves the solution of a weighted closure problem. Replacing the binary search by parametric search [26] would make this algorithm strongly polynomial; however, both search methods depend on the specific properties of the ratio function and would not work for other bicriterion problems. A second paper, by Carlson and Eppstein [6], considers bicriterion versions of the problem of finding the best subtree (containing the root) of a given rooted tree with weighted edges. These subtrees can be modeled as lower sets for a partial order on the tree edges in which two edges are comparable when they both belong to a path from the root; this partial order is series-parallel, and we greatly generalize the results of Carlson and Eppstein in our new results on parametric closures for series-parallel partial orders.

Parametric optimization as an implicit convex hull problem. Parametric optimization problems can be formulated dually, as problems of computing convex hulls of implicitly defined two-dimensional point sets. Suppose we are given a parametric optimization problem in which weight of element i is a linear function $a_i\lambda + b_i$ of a parameter λ, and in which the weight of a candidate solution S (a subset of elements, constrained by the specific optimization problem in question) is the sum of these functions. Then the solution value is also a linear function, whose coefficients are the sums of the element coefficients:

$$\sum_{i\in S} a_i\lambda + b_i = \left(\sum_{i\in S} a_i\right)\lambda + \left(\sum_{i\in S} b_i\right).$$

Instead of interpreting the numbers a_i and b_i as coefficients of linear functions, we may re-interpret the same two numbers as the x and y coordinates (respectively) of points in the Euclidean plane. In this way any family $\mathcal{F}$ of candidate solutions determines a planar point set, in which each set in $\mathcal{F}$ corresponds to the point given by the sum of its elements' coefficients. We call this point set $\mathsf{project}(\mathcal{F})$, because the sets in $\mathcal{F}$ can be thought of as vertices of a hypercube $Q_n = \{0,1\}^n$ whose dimension is the number of input elements, and $\mathsf{project}$ determines a linear projection from these vertices to the Euclidean plane.

Let $\mathsf{hull}(\mathsf{project}(\mathcal{F}))$ denote the convex hull of this projected planar point set. Then for each parameter value the set in $\mathcal{F}$ minimizing or maximizing the parameterized weight corresponds by projective duality to a vertex of the hull,

and the same is true for the maximizer of any quasiconvex function of the two sums of coefficients a_i and b_i. Thus, parametric optimization can be reformulated as the problem of constructing this convex hull, and bicriterion optimization can be solved by choosing the best hull vertex.

New results. For an arbitrary partially ordered set P, define $\mathsf{lower}(P)$ to be the family of lower sets of P. As a convenient abbreviation, we define $\mathsf{polygon}(P) = \mathsf{hull}(\mathsf{project}(\mathsf{lower}(P)))$. We consider the following classes of partially ordered set. For each partial order P in one of these classes, we prove polynomial bounds on the complexity of $\mathsf{polygon}(P)$ and on the time for constructing the hull. These results imply the same time bounds for parametric optimization over P and for maximizing a quasiconvex function over P.

Semiorders. This class of partial orders was introduced to model human preferences [24] in which each element can be associated with a numerical value, pairs of elements whose values are within a fixed margin of error are incomparable, and farther-apart pairs are ordered by their numerical values. For such orderings, we give a bound of $O(n \log n)$ on the complexity of $\mathsf{polygon}(P)$ and we show that it can be constructed in time $O(n \log^2 n)$ using an algorithm based on the quadtree data structure.

Series-parallel partial orders. These are orders formed recursively from smaller orders of the same type by two operations: series compositions (in which all elements from one order are placed earlier in the combined ordering than all elements of the other order) and parallel compositions (in which pairs of one element from each ordering are incomparable). These orderings have been applied for instance in scheduling applications by Lawler [22]. For such orderings, the sets of the form $\mathsf{polygon}(P)$ have a corresponding recursive construction by two operations: the convex hull of the union of two convex polygons, and the Minkowski sum of two convex polygons. It follows that $\mathsf{polygon}(P)$ has complexity $O(n)$. This construction does not immediately lead to a fast construction algorithm, but we adapt a splay tree data structure to construct $\mathsf{polygon}(P)$ in time $O(n \log n)$. Our previous results for optimal subtrees [6] follow as a special case of this result.

Bounded treewidth. Suppose that partial order P has n elements and its transitive reduction forms a directed acyclic graph whose underlying undirected graph has treewidth w. (For prior work on treewidth of partial orders, see [20].) Then we show that $\mathsf{polygon}(P)$ has polynomially many vertices, with exponent $O(w)$, and that it can be constructed in polynomial time.

Incidence posets The incidence poset of a graph G has the vertices and edges as elements, with an order relation $x \leq y$ whenever x is an endpoint of y. One of the initial applications for the closure problem concerned the design of freight delivery systems in which a certain profit could be expected from each of a set of point-to-point routes in the system, but at the cost of setting up depots at each endpoint of the routes [3, 30]; this can be modeled with an incidence poset for a graph with a vertex at each depot location and an edge for each potential route. Since the profits and costs have different

timeframes, it is reasonable to combine them in a nonlinear way, giving a bicriterion closure problem. The transitive reduction of an incidence poset is a subdivision of G with the same treewidth, so our technique for partial orders of bounded treewidth also applies to incidence posets of graphs of bounded treewidth.

Bounded width. The *width* of a partial order is the maximum number of elements in an *antichain*, a set of mutually-incomparable elements. Low-width partial orders arise, for instance, in the edit histories of version control repositories [4]. The treewidth of a partial order is at most equal to its width, but partial orders of width w have $O(n^w)$ lower sets, tighter than the bound that would be obtained by using treewidth. We show more strongly using quadtrees that in this case $\mathsf{polygon}(P)$ has $O(n^{w-1} + n \log n)$ vertices and can be constructed in time within a logarithmic factor of this bound.

We have been unable to obtain an example of a family of partial orders with a nonlinear lower bound on the complexity of $\mathsf{polygon}(P)$, nor have we been able to obtain a nontrivial upper bound on the hull complexity for unrestricted partial orders. Additionally, we have been unable to obtain polynomial bounds on the hull complexity of the above types of partial orders for dimensions higher than two. We also do not know of any computational complexity bounds (such as NP-hardness) for the parametric closure problem for any class of partial orders in any finite dimension. We leave these problems open for future research.

For space reasons we describe only the semiorder and series-parallel results in the main text of our paper, deferring the remaining results to appendices.

2 Minkowski sums and hulls of unions

Our results on the complexity of the convex polygons $\mathsf{polygon}(P)$ associated with a partial order hinge on decomposing these polygons recursively into combinations of simpler polygons. To do this, we use two natural geometric operations that combine pairs of convex polygons to produce more complex convex polygons.

Definition 1. *For any two convex polygons P and Q, let $P \oplus Q$ denote the Minkowski sum of P and Q (the set of points that are the vector sum of a point in P and a point in Q), and let $P \Cup Q$ denote the convex hull of the union of P and Q.*

Lemma 1 (folklore). *If convex polygons P and Q have p and q vertices respectively, then $P \oplus Q$ and $P \Cup Q$ have at most $p + q$ vertices, and can be constructed from P and Q in time $O(p + q)$.*

We omit the (easy) proof for space reasons.

Corollary 1. *Suppose that P is a convex polygon, described as a formula that combines a set of n points in the plane into a single polygon using a sequence of $\oplus$ and $\Cup$ operations. Suppose in addition that, when written as an expression tree, this formula has height h. Then P has at most n vertices and it may be constructed from the formula in time $O(nh)$.*

More complex data structures can reduce this time to $O(n \log n)$; see Section 4.

In higher dimensions, the convex hull of n points and Minkowski sum of n line segments both have polynomial complexity with an exponent that depends linearly on the dimension. However, we do not know of an analogous bound on the complexity of convex sets formed by combining Minkowski sum and hull-of-union operations. If such a bound held, we could extend our results on parametric closures to the corresponding higher dimensional problems.

3 Semiorders

A *semiorder* is a type of partial order defined by Luce [24] to model human preferences. Each element of the order has an associated numerical value (its *utility* to the person whose preferences are being modeled). For items whose utilities are sufficiently far from each other, the ordering of the two items in the semiorder is the same as the numerical ordering of their utilities. However, items whose utilities are within some (global) margin of error of each other are incomparable in the semiorder. Similar concepts of comparisons of numerical values with margins of error give rise to semiorders in many other areas of science and statistics [29]. For efficient computations on semiorders we will assume that the utility values of each element are part of the input to an algorithm, and that the margin of error has been normalized to one. For instance, the semiorder N of Figure 1 can be represented as a semiorder with utilities 2/3, 0, 2, and 4/3 for a, b, c, and d respectively. With this information in hand, the comparison between any two elements can be determined in constant time.

The concept of a lower set is particularly natural for a semiorder: it is a set of elements whose utility values could lie below a sharp numerical threshold, after perturbing each utility value by at most half the margin of error. In this way, the closure problem (the problem of finding a maximum weight lower set) can alternatively be interpreted as the problem of finding the maximum possible discrepancy of a one-dimensional weighted point set in which the location of each point is known imprecisely. Semiorders may have exponentially many lower sets; for instance, if all items have utilities that are within one unit of each other, all sets are lower sets. Nevertheless, as we show in this section, if S is a semiorder, then the complexity of $\mathsf{polygon}(S)$ is near-linear.

If S is any parametrically weighted semi-order, we may write the sorted order of the utility values of elements of S as $u_0, u_1, \ldots, u_{n-1}$ where $n = |S|$, and the elements themselves (in the same order) as $x_0, x_1, \ldots, x_{n-1}$. By padding S with items that have a fixed zero weight and a utility that is smaller than that of the elements by more than the margin of error, we may assume without loss of generality that n is a power of two without changing the values of the parametric closure problem on S.

Definition 2. *Let L be an arbitrary lower set in $\mathsf{lower}(S)$. Let j be the largest index of an element x_i of L. Let i be the smallest index of an element x_i such that x_i does not belong to L and $i < j$, or -1 if no such element exists. Define $\mathsf{extremes}(L)$ to be the pair of integers $(i + 1, j)$.*

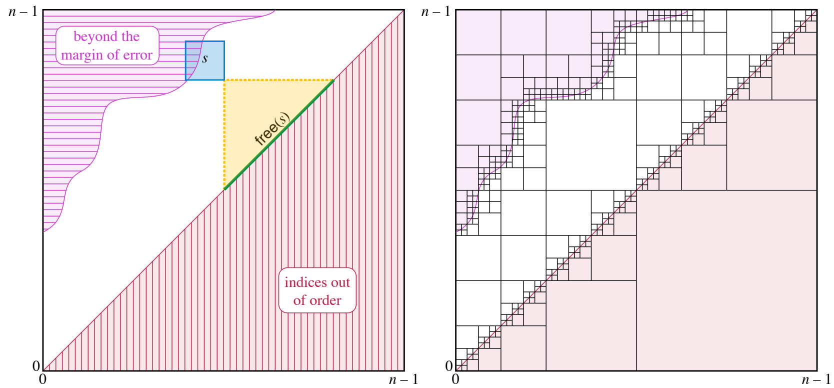

Fig. 2. The grid $[0, n-1]^2$, with the two regions that cannot be part of the image of extremes. The left image shows a square subproblem s and free(s); the right image shows the quadtree decomposition of the grid used to prove Theorem 1.

Thus, extremes maps the family lower(S) to the integer grid $[0, n-1]^2$, with potentially many lower sets mapped to each grid point. However, not every grid point is in the image of lower(S): a point (i, j) with $i > j$ cannot be the image of a lower set, because the element defining the first coordinate of extremes must have an index smaller than the element defining the second coordinate. And when $i > 0$, a point (i, j) with $u_{i-1} < u_j - 1$ (i.e. with utility values that are beyond the margin of error for the semiorder) also cannot be the image of a lower set, because in this case $x_{i-1} \leq x_j$ in the semiorder, so every lower set that includes x_j also includes x_{i-1}. Thus, the image of extremes lies in an orthogonally convex subset of the grid, bounded below by its main diagonal and above by a monotone curve (Figure 2).

Definition 3. *Let s be any square subset of the integer grid $[0, n-1]^2$, and define* subproblem(s) *to be the partially-ordered subset of the semiorder S consisting of the elements whose indices are among the rows and columns of s. Define* free(s) *to be the (unordered) set of elements of S that do not belong to* subproblem(s)*, but whose indices are between pairs of indices that belong to* subproblem(s)*. (See Figure 2, left, for an example.)*

Observation 1 *Given a square s, suppose that the subfamily $\mathcal{F}$ of* lower(S) *that is mapped by* extremes *to s is nonempty. Then each set in $\mathcal{F}$ is the disjoint union of a lower set of* subproblem(s) *and an arbitrary subset of* free(s)*, and all such disjoint unions belong to $\mathcal{F}$.*

Observation 2 *For any square s, let* powerset(free(s)) *be the family of all subsets of* free(s)*. Let weight function $w : S \to \mathbb{R}^2$ define a projection* project *from families of sets to point sets in $\mathbb{R}^2$. Then* project(powerset(free(s))) *is the Minkowski sum of the sets $\{(0, 0), w(x_i)\}$ for $x_i \in$* free(s)*. Its convex hull is a centrally symmetric convex polygon* hull(project(powerset(free(s)))) *(the Minkowski sum of*

the corresponding line segments) with at most $k = 2|\mathsf{free}(s)|$ sides, and can be constructed in time $O(k \log k)$.

Corollary 2. *Given a square s, let $\mathcal{F}$ be the sub-family of $\mathsf{lower}(S)$ that is mapped by $\mathsf{extremes}$ into s, and assume a weight function w defining a projection $\mathsf{project}$. Then $\mathsf{hull}(\mathsf{project}(\mathcal{F})) = \mathsf{polygon}(\mathsf{subproblem}(s)) \oplus \mathsf{hull}(\mathsf{project}(\mathsf{powerset}(\mathsf{free}(s))))$.*

Lemma 2. *Let s be a square in the grid $[0, n-1]^2$, and subdivide s into four congruent smaller squares s_i ($0 \le i < 4$). Let $\mathsf{polygon}(\mathsf{subproblem}(s))$ have c vertices, define c_i in the same way for each s_i, and let ℓ be the side length of s. Then $c \le \sum c_i + O(\ell)$, and $\mathsf{polygon}(\mathsf{subproblem}(s))$ can be constructed from the corresponding hulls for the smaller squares in time $O(\sum c_i + \ell \log \ell)$.*

Proof. For each smaller square s_i, define $\mathcal{F}_i$ to be the subset of $\mathsf{subproblem}(s)$ mapped to s_i, and define $H_i = \mathsf{hull}(\mathsf{project}(\mathcal{F}_i))$. Then by Corollary 2 (viewing s_i as a subproblem of $\mathsf{subproblem}(s)$),

$$H_i = \mathsf{polygon}(\mathsf{subproblem}(s)) \oplus \mathsf{hull}(\mathsf{project}(\mathsf{powerset}(\mathsf{free}(s) \setminus \mathsf{free}(s_i)))).$$

The set $\mathsf{free}(s) \setminus \mathsf{free}(s_i)$ has cardinality $O(\ell)$, so by Lemma 1 and Observation 2, H_i has complexity $c_i + O(\ell)$ and can be constructed in time $O(c_i + \ell \log \ell)$. Applying Lemma 1 again, $\mathsf{polygon}(\mathsf{subproblem}(s)) = H_0 \Cup H_1 \Cup H_2 \Cup H_2$ has complexity at most $\sum c_i + O(\ell)$ and can be constructed from the polygons H_i in time $O(\sum c_i + \ell \log \ell)$. ◻

Theorem 1. *If S is a semiorder with n elements x_i, specified with their utility values u_i and a system of two-dimensional weights $w(x_i)$, then $\mathsf{polygon}(S)$ has complexity $O(n \log n)$ and can be constructed in time $O(n \log^2 n)$.*

Proof. We sort the utility values, pad n to the next larger power of two if necessary and form a quadtree decomposition of the grid $[0, n-1]^2$ (as shown in Figure 2, right). For each square s of this quadtree, we associate a convex polygon (or empty set) $\mathsf{polygon}(\mathsf{subproblem}(s))$ computed according to the following cases:

- If s is a subset of the grid points for which $i > j$, or for which $u_{i-1} < u_j - 1$, then no lower sets are mapped into s by $\mathsf{extremes}$. We associate square s with the empty set.
- If s is a subset of the grid points for which $i \le j$ and $u_{i-1} \ge u_j - 1$, then every two elements of $\mathsf{subproblem}(s)$ are incomparable. In this case, we associate square s with the polygon $\mathsf{hull}(\mathsf{project}(\mathsf{powerset}(\mathsf{subproblem}(s))))$ computed according to Observation 2.
- Otherwise, we split s into four smaller squares. We construct the polygon associated with s by using Lemma 2 to combine the polygons associated with its children.

It follows by induction that the total complexity of the polygon constructed at any square s of the quadtree is $O(\sum \ell_i)$, and the total time for constructing it is $O(\sum \ell_i \log \ell_i)$, where ℓ_i is the side length of the ith square of the quadtree

and the sum ranges over all descendants of s. As a base case for the induction, a square containing only a single grid point is associated with a subproblem with one element, with only one lower set that maps to that grid point, and a degenerate convex polygon with a single vertex. The polygon constructed at the root of the quadtree is the desired output, and it follows that it has combinatorial complexity and time complexity of the same form, with a sum ranging over all quadtree squares.

The conditions $i > j$ and $u_{i-1} < u_j - 1$ define two monotone curves through the grid, and we split a quadtree square only when it is crossed by one of these two curves. It follows that the squares of side length ℓ that are subdivided as part of this algorithm themselves form two monotone chains, and that the number of all squares of side length ℓ is $O(n/\ell)$. The results of the theorem follow by summing up the contributions to the polygon complexity and time complexity for the $O(\log n)$ different possible values of ℓ. $\square$

4 Series-parallel partial orders

Series-parallel partial orders were considered in the context of a scheduling problem by Lawler [22], and include as a special case the tree orderings previously studied in our work on bicriterion optimization [6]. They are the partial orders that can be constructed from single-element partial orders by repeatedly applying the following two operations:

Series composition. Given two series-parallel partial orders P_1 and P_2, form an order from their disjoint union in which every element of P_1 is less than every element of P_2.

Parallel composition. Given two series-parallel partial orders P_1 and P_2, form an order from their disjoint union in which there are no order relations between P_1 and P_2.

Observation 3 *If P is the series composition of P_1 and P_2, then* $\mathsf{polygon}(P)$ *is the convex hull of the union of* $\mathsf{polygon}(P_1)$ *and a translate (by the sum of the weights of the elements of P_1) of* $\mathsf{polygon}(P_2)$. *If P is the parallel composition of P_1 and P_2, then* $\mathsf{polygon}(P)$ *is the Minkowski sum of* $\mathsf{polygon}(P_1)$ *and* $\mathsf{polygon}(P_2)$.

Recursively continuing this decomposition gives us a formula for $\mathsf{polygon}(P)$ in terms of the $\Cup$ and *oplus* operations. By Lemma 1 we immediately obtain:

Corollary 3. *If P is a series-parallel partial order with n elements, then* $\mathsf{polygon}(P)$ *has at most $2n$ vertices.*

However, the depth of the formula for $\mathsf{polygon}(P)$ may be linear, so using Lemma 1 to construct $\mathsf{polygon}(P)$ could be inefficient. We now describe a faster algorithm. The key idea is to follow the same formula to build $\mathsf{polygon}(P)$, but to represent each intermediate result (a convex polygon) by a data structure that allows the $\Cup$ and $\oplus$ operations to be performed more quickly for pairs of

polygons of unbalanced sizes. Note that a Minkowski sum operation between a polygon of high complexity and a polygon of bounded complexity can change a constant fraction of the vertex coordinates, so to allow fast Minkowski sums our representation cannot store these coordinates explicitly.

Lemma 3. *It is possible to store convex polygons in a data structure such that destructively merging the representations of two polygons of m and n vertices respectively by a $\Cup$ or $\oplus$ operation (with $m < n$) can be performed in time $O(m \log((m+n)/m))$.*

Proof. We store the lower and upper hulls separately in a binary search tree data structure, in which each node represents a vertex of the polygon, and the inorder traversal of the tree gives the left-to-right order of the vertices. The node at the root of the tree stores the Cartesian coordinates of its vertex; each non-root node stores the vector difference between its coordinates and its parents' coordinates. Additionally, each node stores the vector difference to its clockwise neighbor around the polygon boundary. In this way, we can traverse any path in this tree and (by adding the stored vector difference) determine the coordinates of any vertex encountered along the path. We may also perform a rotation in the tree, and update the stored vector differences, in constant time per rotation.

We will keep this tree balanced (in an amortized sense) by using the splay tree balancing strategy [31]: whenever we follow a search path in the tree, we will immediately perform a splay operation that through a sequence of double rotations moves the endpoint of the path to the root of the tree. By the dynamic finger property for splay trees [9, 10], a sequence of m accesses in sequential order into a splay tree of size n will take time $O(m \log((m+n)/m))$.

To compute the hull of the union (the $\Cup$ operation) we insert each vertex of the smaller polygon (by number of vertices), in left-to-right order, into the larger polygon. To insert a vertex v, we search the larger polygon to find the edges with the same x-coordinate as v and use these edges to check whether v belongs to the lower hull, the upper hull, or neither. If it belongs to one of the two hulls, we search the larger polygon again to find its two neighbors on the hull. By performing a splay so that these neighbors are rotated to the root of the binary tree, and then cutting the tree at these points, we may remove the vertices between v and its new neighbors from the tree without having to consider those vertices one-by-one. We then create a new node for v and add its two neighbors as the left and right child.

To compute the Minkowski sum (the $\oplus$ operation) we must simply merge the two sequences of edges of the two polygons by their slopes. We search for each edge slope in the smaller polygon. When its position is found, we splay the vertex node at the split position to the root of the tree, and then split the tree into its left and right subtrees, each with a copy of the root node. We translate all vertices on one side of the split by the vector difference for the inserted edge (by adding that vector only to the root of its tree), and rejoin the trees. $\square$

Theorem 2. *If P is a series-parallel partial order, represented by its series-parallel decomposition tree, then* $\mathsf{polygon}(P)$ *has complexity $O(n)$ and may be constructed in time $O(n \log n)$.*

Proof. We follow the formula for constructing $\mathsf{polygon}(P)$ by $\Cup$ and $\oplus$ operations, using the data structure of Lemma 3. We charge each merge operation to the partial order elements on the smaller side of each merge. If a partial order element belongs to subproblems of sizes $n_0 = 1, n_1, \ldots, n_h = n$ where h is the height of the element, then the time charged to it is $\sum_i O(\log(n_i/n_{i-1})) = O(\log \prod_i (n_i/n_{i-1})) = O(\log n)$. $\square$

Acknowledgements. Supported in part by NSF grant 1228639 and ONR grant N00014-08-1-1015.

References

1. P. K. Agarwal, D. Eppstein, L. J. Guibas, and M. R. Henzinger. Parametric and kinetic minimum spanning trees. *Proc. 39th IEEE Symp. Foundations of Computer Science (FOCS '98)*, pp. 596–605, 1998, doi:10.1109/SFCS.1998.743510.
2. M. Ahmed, I. Chowdhury, M. Gibson, M. S. Islam, and J. Sherrette. On maximum weight objects decomposable into based rectilinear convex objects. *Proc. 13th Int. Symp. Algorithms and Data Structures (WADS 2013)*, pp. 1–12. Springer, Lect. Notes Comput. Sci. 8037, 2013, doi:10.1007/978-3-642-40104-6_1.
3. M. L. Balinski. On a selection problem. *Manag. Sci.* 17(3):230–231, 1970, doi:10.1287/mnsc.17.3.230.
4. M. J. Bannister, W. E. Devanny, and D. Eppstein. Small superpatterns for dominance drawing. *Proc. Analytic Algorithmics and Combinatorics (ANALCO14)*, pp. 92–103, 2014, doi:10.1137/1.9781611973204.9.
5. K. Buchin, D. Eppstein, M. Löffler, and R. I. Silveira. Adjacency-preserving spatial treemaps. *Proc. 11th Int. Symp. Algorithms and Data Structures (WADS 2011)*, pp. 159–170. Springer, Lect. Notes Comput. Sci. 6844, 2011, doi:10.1007/978-3-642-22300-6_14.
6. J. Carlson and D. Eppstein. The weighted maximum-mean subtree and other bicriterion subtree problems. *Proc. 10th Scand. Worksh. Algorithm Theory (SWAT 2006)*, pp. 397–408. Springer, Lect. Notes Comput. Sci. 4059, 2006, doi:10.1007/11785293_37.
7. P. J. Carstensen. Parametric cost shortest path problems. Unpublished memo, Bellcore, 1984.
8. G. J. Chang and J. Edmonds. The poset scheduling problem. *Order* 2(2):113–118, 1985, doi:10.1007/BF00334849.
9. R. Cole. On the dynamic finger conjecture for splay trees. II. The proof. *SIAM J. Comput.* 30(1):44–85, 2000, doi:10.1137/S009753979732699X.
10. R. Cole, B. Mishra, J. Schmidt, and A. Siegel. On the dynamic finger conjecture for splay trees. I. Splay sorting $\log n$-block sequences. *SIAM J. Comput.* 30(1):1–43, 2000, doi:10.1137/S0097539797326988.
11. R. P. Dilworth. A decomposition theorem for partially ordered sets. *Ann. Math.* 51(1):161–166, 1950, doi:10.2307/1969503.
12. D. Eppstein. Geometric lower bounds for parametric matroid optimization. *Discrete Comput. Geom.* 20(4):463–476, 1998, doi:10.1007/PL00009396.

13. B. Faaland, K. Kim, and T. Schmitt. A new algorithm for computing the maximal closure of a graph. *Manag. Sci.* 36(3):315–33, 1990, doi:10.1287/mnsc.36.3.315.
14. D. Fernández-Baca, G. Slutzki, and D. Eppstein. Using sparsification for parametric minimum spanning tree problems. *Nordic J. Comput.* 3(4):352–366, 1996.
15. M. Gibson, D. Han, M. Sonka, and X. Wu. Maximum weight digital regions decomposable into digital star-shaped regions. *Proc. 22nd Int. Symp. Algorithms and Computation (ISAAC 2011)*, pp. 724–733. Springer, Lect. Notes Comput. Sci. 7074, 2011, doi:10.1007/978-3-642-25591-5_74.
16. J. Hershberger. Finding the upper envelope of n line segments in $O(n \log n)$ time. *Information Processing Letters* 33(4):169–174, 1989, doi:10.1016/0020-0190(89)90136-1.
17. D. Hochbaum. A new-old algorithm for minimum-cut and maximum-flow in closure graphs. *Networks* 37(4):171–193, 2001, doi:10.1002/net.1012.
18. D. Hochbaum. 50th anniversary article: Selection, provisioning, shared fixed costs, maximum closure, and implications on algorithmic methods today. *Manag. Sci.* 50(6):709–723, 2004, doi:10.1287/mnsc.1040.0242.
19. R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *J. ACM* 34(3):532–543, 1987, doi:10.1145/28869.28871.
20. G. Joret, P. Micek, K. G. Milans, W. T. Trotter, B. Walczak, and R. Wang. Tree-width and dimension, arXiv:1301.5271. Unpublished manuscript, 2013.
21. N. Katoh. Bicriteria network optimization problems. *IEICE Trans. Fundamentals of Electronics, Communications and Computer Sciences* E75-A:321–329, 1992.
22. E. L. Lawler. Sequencing jobs to minimize total weighted completion time subject to precedence constraints. *Ann. Discrete Math.* 2:75–90, 1978, doi:10.1016/S0167-5060(08)70323-6.
23. H. Lerchs and I. F. Grossmann. Optimum design of open-pit mines. *Trans. Canad. Inst. Mining and Metallurgy* 68:17–24, 1965.
24. R. D. Luce. Semiorders and a theory of utility discrimination. *Econometrica* 24:178–191, 1956, doi:10.2307/1905751.
25. J. Matoušek. Lower envelopes. *Lectures on Discrete Geometry*, pp. 165–194. Springer, Graduate Texts in Mathematics 212, 2002, `10.1007/978-1-4613-0039-7_7`.
26. N. Megiddo. Applying parallel computation algorithms in the design of serial algorithms. *J. ACM* 30(4):852–865, 1983, doi:10.1145/2157.322410.
27. D. Orlin. Optimal weapons allocation against layered defenses. *Nav. Res. Logist. Q.* 34:605–617, 1987, doi:10.1002/1520-6750(198710)34:5<605::aid-nav3220340502>3.0.co;2-l.
28. J.-C. Picard. Maximal closure of a graph and applications to combinatorial problems. *Manag. Sci.* 22(11):1268–1272, 1976, doi:10.1287/mnsc.22.11.1268.
29. M. Pirlot and P. Vincke. *Semiorders: Properties, Representations, Applications.* Springer, 1997.
30. J. M. W. Rhys. A selection problem of shared fixed costs and network flows. *Manag. Sci.* 17(3):200–207, 1970, doi:10.1287/mnsc.17.3.200.
31. D. D. Sleator and R. E. Tarjan. Self-adjusting binary search trees. *J. ACM* 32(3):652–686, 1985, doi:10.1145/3828.3835.

A Bounded tree-width

In this section, we consider partial orders defined by reachability on directed acyclic graphs whose underlying undirected graph has bounded treewidth.

A *tree-decomposition* of an undirected graph G is a tree T whose vertices are called *bags*, each of which is associated with a set of vertices of G, with the following two properties:

- Each vertex of G belongs to a set of bags that induces a connected subtree of T, and
- For each edge of G there exists a bag containing both endpoints of the edge.

The *width* of a tree-decomposition is one less than the maximum number of vertices in a bag, and the *treewidth* of G is the minimum width of a tree-decomposition of G. The treewidth, and an optimal tree-decomposition, can be found in time linear in the number of vertices of a given graph and exponential in its width.

By splitting nodes with high degree into more than one node (all associated with equal sets of vertices of G), we may assume without loss of generality that T has maximum degree three. We may also assume without loss of generality that T has $O(n)$ nodes. These assumptions do not change the width of the decomposition.

We will use the following well-known result:

Lemma 4 (folklore). *Any k-node forest F of maximum degree three can be partitioned by the deletion of one node x into two smaller forests, each having at most $2k/3$ nodes.*

Proof. Let x be chosen to minimize the maximum size of a tree in the remaining forest. Then each tree has at most $2k/3$ nodes: otherwise we could reduce the maximum tree size by moving x into the largest tree or (if it is already there) moving x to one of its neighbors.

We use a greedy strategy to select trees for one of the two forests, largest first, until the forest has at least $k/3$ nodes. If this happens on the first step, we are done; otherwise, each step increases the size of the forest by less than $k/3$ nodes, so the size of the forest cannot jump from less than $k/3$ to greater than $2k/3$ at any step. □

Lemma 5. *Let P be a partial order with n elements whose underlying graph has treewidth w. Then $\mathsf{polygon}(P)$ can be represented as a formula of binary $\Cup$ and $\oplus$ operations with height at most $(w+2)(\log_{3/2} n + O(1))$.*

Proof. We recursively decompose the given tree decomposition. At each step of the decomposition we will have a subforest F of the original tree decomposition (initially T itself), a bag B of F chosen according to Lemma 4 (initially, the choice given by the lemma for T), and an assignment of some elements of P to two disjoint subsets L and U of P (initially empty). The meaning of this information is that we restrict our attention to lower sets on the elements of P that participate in bags of F, and that are consistent with including the elements

of L in the lower set and excluding the elements of U from the lower set. Each subproblem is associated with a polygon, the convex hull of the projections of the lower sets it represents. If an element of L is greater than an element of U, then there are no consistent lower sets and the associated polygon is empty.

Given this information, suppose that some element b of B does not already belong to $L \cup U$. In this case, we partition the problem into two subproblems: one in which b has been added to L and the other in which b has been added to U. The polygon representing the problem prior to this choice of which set to place b into can be represented as the convex hull of the union of the polygons for these two subproblems, because they represent disjoint families of lower sets that together cover all the lower sets represented by the whole problem.

Alternatively, suppose that the elements of bag B are completely covered by $L \cup U$. In this case, let F_1 and F_2 be the two subforests of F, of size smaller by a factor of $2/3$ or better, guaranteed to exist by Lemma 4. We can form subproblems for F_1 and F_2 by using Lemma 4 to choose a bag within each of these two forests. Because all of the elements of bag B have already been placed into L or into U, all remaining choices about the membership of the lower sets in F_1 and in F_2 are completely independent of each other. Therefore, the polygon representing the given problem can be represented as the Minkowski sum of the polygons representing F_1 and F_2.

There can be at most $\log_{3/2} n + O(1)$ steps of this recursive decomposition at which we split the current forest F into two smaller subforests, because each such step reduces the forest size by a $2/3$ or better factor. Between any two such steps there can be at most $w + 1$ steps at which we choose whether to place a vertex of the currently chosen bag into L or U, because there are at most $w + 1$ vertices to place (fewer if the bag is smaller than the maximum bag size or shares vertices with a previously chosen bag). Therefore, the total height of the decomposition is as stated. $\square$

By choosing the initial bag for each subproblem more carefully, we can also represent $\mathsf{polygon}(P)$ as an acyclic circuit of $\Cup$ and $\oplus$ operations with $O(2^w n)$ complexity and $O(w \log n)$ depth (but with a somewhat higher constant factor in the depth). However, we omit the details as we have been unable to generalize Corollary 1 from formulas to circuits.

Theorem 3. *Let P be a partial order with n elements whose underlying graph has treewidth w. Then $\mathsf{polygon}(P)$ has complexity $n^{O(w)}$ and can be constructed in time $n^{O(w)}$.*

Proof. By Lemma 5, there exists a formula of height $(w + 2)(\log_{3/2} n + O(1))$ for $\mathsf{polygon}(P)$. Because this is a formula of binary operations, the total size of the formula is

$$2^{(w+2)(\log_{3/2} n + O(1))} = n^{O(w)}.$$

The result follows by applying Corollary 1 to this formula. $\square$

Corollary 4. *Let P be the incidence poset of an n-vertex graph of treewidth w. Then $\mathsf{polygon}(P)$ has complexity $n^{O(w)}$ and can be constructed in time $n^{O(w)}$.*

For generalized fences and polytrees, we obtain a polynomial bound on the complexity of $\mathsf{polygon}(P)$ by applying Theorem 3 with $w = 1$, but we can tighten this bound in three ways. First, by applying the recursive decomposition to the given tree rather than to its tree decomposition, we need only choose whether to place into L or U a single element for each tree node, rather than the two elements of a bag. Second, we observe that there are only $O(n)$ sequences of choices of which of two forests to continue the recursion in, because each such sequence ends at a single tree node; thus, we can charge all of these choices to a single $O(n)$ factor in the complexity rather than charging a factor of two for each level of formula depth given by these choices. And finally, when the underlying polytree is a path (as it is in the fences and in some definitions of generalized fences) we may always split each subproblem at the path midpoint, reducing the $\log_{3/2} n$ terms in the previous analysis to $\log_2 n$. Based on these observations, we obtain:

Corollary 5. *Let P be the reachability order of a polytree T. Then* $\mathsf{polygon}(P)$ *has complexity $O(n^{1+1/\log_2 3/2})$ and can be constructed in time $O(n^{1+1/\log_2 3/2} \log n)$. If T is an oriented path,* $\mathsf{polygon}(P)$ *has complexity $O(n^2)$ and can be constructed in time $O(n \log n)$.*

B Bounded width

The *width* of a partially ordered set is the size of its largest independent set. Partially ordered sets of bounded width arise, for instance, in the version histories of a distributed version control repository controlled by a small set of developers; in this application, there may be many elements of the partially ordered set (versions of the repository) but the width is bounded by the number of developers [4]. A lower set in this application is a set of versions that could possibly describe the simultaneous states of all the developers at some past moment in the history of the repository.

The lower sets of a partially ordered set correspond one-for-one with its independent sets: each lower set is uniquely determined by an independent set, the set of maximal elements of the lower set. Therefore, in a partially ordered set of width w with n elements there can be at most $O(n^w)$ lower sets. More precisely, by Dilworth's theorem [11], every partial order of width w can be partitioned into w totally-ordered subsets (chains); the number of lower sets can be at most the product of the numbers of lower sets of these chains, $(n/w+1)^w$. As we show, however, there is a tighter bound for the complexity of $\mathsf{polygon}(P)$.

We consider first the case $w = 2$. The more general case of bounded width will follow by similar reasoning. Thus, let P be a partially ordered set with n elements and width two. By Dilworth's theorem, P can be decomposed into two chains, and every lower set L can be described by a pair of integer coordinates (x, y), where x is the number of elements of L that belong to the first chain and y is the number of elements of L in the second chain.

Observation 4 *For a width-two partial order P as described above, the set of pairs of coordinates (x, y) describing the lower sets of P forms an orthogonally convex subset of the integer grid $[0, n]^2$, and the edges between points one unit apart in this set of grid points form the covering relation of the distributive lattice of lower sets.*

Figure 3 depicts an example. We can use the following definition and observation to partition the parametric closure problem for P into smaller subproblems of the same type.

Definition 4. *Let R be any grid rectangle. Then* $\mathsf{subproblem}(R)$ *is the family of lower sets of P whose grid points lie in R.*

Observation 5 *For any grid rectangle, each set in* $\mathsf{subproblem}(R)$ *can be represented uniquely as a disjoint union $A \cup B$ where A is the family of all elements of P whose coordinate value is below or to the left of R, and B is a lower set of the restriction of P to the elements whose coordinate value is within R.*

Observation 6 *Suppose that every integer point of a grid rectangle R corresponds to a lower set of P. Then the restriction of P to the elements whose coordinate value is within R is a partial order in the form of the parallel composition of two chains, a special case of a series-parallel partial order. It follows*

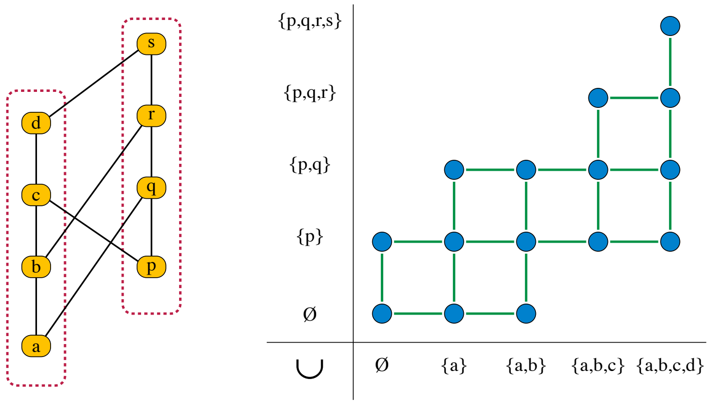

Fig. 3. Hasse diagram of a width-two partial order partitioned into two chains (left) and the subset of the integer lattice formed by its lower sets (right)

from Corollary 3 that $\mathsf{hull}(\mathsf{project}(\mathsf{subproblem}(R)))$ *has complexity proportional to the perimeter of* R.

Theorem 4. *If* P *is a partial order of width two, then* $\mathsf{polygon}(P)$ *has complexity* $O(n \log n)$ *and can be constructed from the covering relation of* P *in time* $O(n \log^2 n)$.

Proof. From the covering relation, we can determine a partition into two chains and trace out the boundaries of the orthogonal grid polygon describing the lower sets of P, in linear time. We use a quadtree to partition the lower sets of P into squares, each with a side length that is a power of two, such that each grid point within each square corresponds to one of the lower sets of P. The squares of each size form two monotone chains within the grid, so the total perimeter of all of these squares is $O(n \log n)$.

For each of the power-of-two subintervals I of each of the two grid coordinates, we compute the convex hull of the lower sets of the corresponding chain whose grid points lie within I, as the hull of the union of the two previously-constructed polygons for the two halves of I. This computation takes time $O(n \log n)$ overall. After this step, each quadtree square's polygon can be constructed as the Minkowski sum of the polygons for its two defining intervals, in time proportional to the side of the square. Therefore, the total time to compute the polygons for all of the quadtree squares is $O(n \log n)$.

It remains to form the convex hull of the union of these polygons. As the convex hull of $O(n \log n)$ points, this takes time $O(n \log^2 n)$. $\square$

For higher widths w, the same idea works (using an octree in three dimensions, etc). The total complexity of $\mathsf{polygon}(P)$ is proportional to the sum of side lengths of octree squares, $O(n^{w-1})$, and the construction time is within a logarithmic factor of this bound.